Amendments to th Claims

Claim 1 (currently amended): A document encoded in an extensible machine-oriented structured notation, wherein the document resides on one or more computer-readable media and comprises:

a node count representing a count of nodes in the document;

a node specification for each of the nodes, each of the node specifications comprising:

a node name;

a child list specifying index values of zero or more nodes which are children of the node;

an attribute list specifying zero or ~~more (attribute name, attribute value) pair~~ more attribute pair references for attributes of the node, each attribute pair reference comprising an attribute name and an attribute value; and

a node value specification, which is empty if the node has no value; and

a data buffer containing attribute names and attribute values referenced from the attribute lists and node values referenced from the node value specifications.

Claim 2 (currently amended): The document according to Claim 1, wherein each attribute pair ~~each (attribute name, attribute value) pair~~ reference specifies a starting name position, a name length, a starting value position, and a value length.

Claim 3 (original): The document according to Claim 2, wherein the starting name position and starting value position are relative to a beginning of the data buffer.

Claim 4 (original): The document according to Claim 2, wherein the starting name position and starting value position are relative to a beginning of the document.

Claim 5 (original): The document according to Claim 1, wherein the node value specification specifies a starting value position and a value length.

Claim 6 (original): The document according to Claim 5, wherein the starting value position is relative to a beginning of the data buffer.

Claim 7 (currently amended): The document according to Claim 5, wherein the ~~starting name position and starting~~ value position is [[are]] relative to a beginning of the document.

Claim 8 (currently amended): The document according to Claim 1, wherein ~~each (attribute name, attribute value) pair~~ each attribute pair reference specifies a starting name position, an ending name position, a starting value position, and an ending value position.

Claim 9 (original): The document according to Claim 1, wherein the node value specification specifies a starting value position and an ending value position.

Claim 10 (currently amended): A computer program product embodied on one or more computer-readable media, the computer program product adapted for encoding a document in an extensible machine-oriented structured notation and comprising:

computer-readable program code means for encoding a node count representing a count of nodes in the document;

computer-readable program code means for encoding a node specification for each of the nodes, further comprising:

computer-readable program code means for encoding a node name;

computer-readable program code means for encoding a child list specifying index values of zero or more nodes which are children of the node;

computer-readable program code means for encoding an attribute list specifying zero or ~~more (attribute name, attribute value) pair~~ more attribute pair references for attributes of the node, each attribute pair reference comprising an attribute name and an attribute value; and

computer-readable program code means for encoding a node value specification, which is empty if the node has no value;

computer-readable program code means for encoding a data buffer containing attribute names and attribute values referenced from the attribute lists and node values referenced from the node value specifications; and

computer-readable program code means for storing the encoded node count, the encoded node specifications, and the encoded data buffer as the encoded document in memory or writing the encoded document to one or more storage media.

Claim 11 (currently amended): A computer program product embodied on one or more computer-readable media, the computer program product adapted for processing a document encoded in an extensible machine-oriented structured notation and comprising:

computer-readable program code means for parsing the document, further comprising:

computer-readable program code means for parsing a node count representing a count of nodes in the document;

computer-readable program code means for parsing a node specification for each of the nodes, further comprising:

computer-readable program code means for parsing a node name;

computer-readable program code means for parsing a child list specifying index values of zero or more nodes which are children of the node;

computer-readable program code means for parsing an attribute list specifying zero or ~~more (attribute name, attribute value) pair~~ more attribute pair references for attributes of the node, each attribute pair reference comprising an attribute name and an attribute value; and

computer-readable program code means for parsing a node value specification, which is empty if the node has no value; and

computer-readable program code means for parsing a data buffer containing attribute names and attribute values referenced from the attribute lists and node values referenced from the node value specifications; and

computer-readable program code means for using the parsed document as input for the processing.

Claim 12 (currently amended): A computer program product embodied on one or more computer-readable media, the computer program product adapted for converting an input

document encoded in an extensible human-friendly extensible markup language ("XML") to an output document encoded in a machine-oriented extensible markup language ("mXML") and comprising:

computer-readable program code means for creating a document tree representation of the input document;

computer-readable program code means for obtaining a node count representing a count of nodes in the document tree representation;

computer-readable program code means for writing the node count to an mXML buffer;

computer-readable program code means for traversing each node in the document tree representation and generating a corresponding node specification in the mXML buffer, further comprising:

computer-readable program code means for generating a node name;

computer-readable program code means for generating an attribute list specifying zero or ~~more (attribute name, attribute value) pair~~ more attribute pair references for attributes of the node, each attribute pair reference comprising an attribute name and an attribute value;

computer-readable program code means for generating a child list specifying index values of zero or more nodes which are children of the node; and

computer-readable program code means for generating a node value specification, which is empty if the node has no value;

computer-readable program code means for generating a data buffer containing attribute names and attribute values referenced from the attribute lists and node values referenced from the node value specifications; and

computer-readable program code means for appending the data buffer to the mXML buffer to form the output document.

Claim 13 (currently amended): The computer program product according to Claim 12, wherein the computer-readable program code means for generating ~~each (attribute name, attribute value)~~ pair each attribute pair reference further comprises computer-readable program code means for generating a starting name position, a name length, a starting value position, and a value length.

Claim 14 (original): The computer program product according to Claim 13, wherein the starting name position and starting value position are relative to a beginning of the data buffer.

Claim 15 (original): The computer program product according to Claim 13, wherein the starting name position and starting value position are relative to a beginning of the output document.

Claim 16 (original): The computer program product according to Claim 12, wherein the node value specification specifies a starting value position and a value length.

Claim 17 (original): The computer program product according to Claim 15, wherein the starting value position is relative to a beginning of the data buffer.

Claim 18 (currently amended): The computer program product according to Claim 15, wherein the ~~starting name position and~~ starting value position [[are]] is relative to a beginning of the

output document.

Claim 19 (currently amended): The computer program product according to Claim 12, wherein the computer-readable program code means for generating ~~each (attribute name, attribute value) pair~~ each attribute pair reference further comprises computer-readable program code means for generating a starting name position, an ending name position, a starting value position, and an ending value position.

Claim 20 (original): The computer program product according to Claim 12, wherein the node value specification specifies a starting value position and an ending value position.

Claim 21 (currently amended): A system for encoding a document in an extensible machine-oriented structured notation, comprising:

means for encoding a node count representing a count of nodes in the document;

means for encoding a node specification for each of the nodes, further comprising:

means for encoding a node name;

means for encoding a child list specifying index values of zero or more nodes which are children of the node;

means for encoding an attribute list specifying zero or ~~more (attribute name, attribute value) pair~~ more attribute pair references for attributes of the node, each attribute pair reference comprising an attribute name and an attribute value; and

means for encoding a node value specification, which is empty if the node has no

value;

means for encoding a data buffer containing attribute names and attribute values referenced from the attribute lists and node values referenced from the node value specifications; and

means for storing the encoded node count, the encoded node specifications, and the encoded data buffer as the encoded document in memory or writing the encoded document to one or more storage media.

Claim 22 (currently amended): A system for processing a document encoded in an extensible machine-oriented structured notation, comprising:

means for parsing the document, further comprising:

means for parsing a node count representing a count of nodes in the document;

means for parsing a node specification for each of the nodes, further comprising:

means for parsing a node name;

means for parsing a child list specifying index values of zero or more nodes which are children of the node;

means for parsing an attribute list specifying zero or ~~more (attribute name, attribute value) pair~~ more attribute pair references for attributes of the node, each attribute pair reference comprising an attribute name and an attribute value; and

means for parsing a node value specification, which is empty if the node has no value; and

means for parsing a data buffer containing attribute names and attribute values

referenced from the attribute lists and node values referenced from the node value specifications; and

means for using the parsed document as input for the processing.

Claim 23 (currently amended): A system for converting an input document encoded in an extensible human-friendly extensible markup language ("XML") to an output document encoded in a machine-oriented extensible markup language ("mXML"), comprising:

means for creating a document tree representation of the input document;

means for obtaining a node count representing a count of nodes in the document tree representation;

means for writing the node count to an mXML buffer;

means for traversing each node in the document tree representation and generating a corresponding node specification in the mXML buffer, further comprising:

means for generating a node name;

means for generating an attribute list specifying zero or ~~more (attribute name, attribute value) pair~~ more attribute pair references for attributes of the node, each attribute pair reference comprising an attribute name and an attribute value;

means for generating a child list specifying index values of zero or more nodes which are children of the node; and

means for generating a node value specification, which is empty if the node has no value;

means for generating a data buffer containing attribute names and attribute values

referenced from the attribute lists and node values referenced from the node value specifications; and

means for appending the data buffer to the mXML buffer to form the output document.

Claim 24 (currently amended): The system according to Claim 23, wherein the means for generating ~~each (attribute name, attribute value) pair~~ each attribute pair reference further comprises means for generating a starting name position, a name length, a starting value position, and a value length.

Claim 25 (original): The system according to Claim 24, wherein the starting name position and starting value position are relative to a beginning of the data buffer.

Claim 26 (original): The system according to Claim 24, wherein the starting name position and starting value position are relative to a beginning of the output document.

Claim 27 (original): The system according to Claim 23, wherein the node value specification specifies a starting value position and a value length.

Claim 28 (original): The system according to Claim 26, wherein the starting value position is relative to a beginning of the data buffer.

Claim 29 (currently amended): The system according to Claim 26, wherein the ~~starting name~~

position and starting value position [[are]] is relative to a beginning of the output document.

Claim 30 (currently amended): The system according to Claim 23, wherein the means for generating ~~each (attribute name, attribute value) pair~~ each attribute pair reference further comprises means for generating a starting name position, an ending name position, a starting value position, and an ending value position.

Claim 31 (original): The system according to Claim 23, wherein the node value specification specifies a starting value position and an ending value position.

Claim 32 (currently amended): A method for encoding a document in an extensible machine-oriented structured notation, comprising the steps of:

encoding a node count representing a count of nodes in the document;

encoding a node specification for each of the nodes, further comprising the steps of:

encoding a node name;

encoding a child list specifying index values of zero or more nodes which are children of the node;

encoding an attribute list specifying zero or more ~~(attribute name, attribute value)~~ pair more attribute pair references for attributes of the node, each attribute pair reference comprising an attribute name and an attribute value; and

encoding a node value specification, which is empty if the node has no value;

encoding a data buffer containing attribute names and attribute values referenced from the

attribute lists and node values referenced from the node value specifications; and

storing the encoded node count, the encoded node specifications, and the encoded data buffer as the encoded document in memory or writing the encoded document to one or more storage media.

Claim 33 (currently amended): A method for processing a document encoded in an extensible machine-oriented structured notation, comprising the steps of:

parsing the document, further comprising the steps of:

parsing a node count representing a count of nodes in the document;

parsing a node specification for each of the nodes, further comprising the steps of:

parsing a node name;

parsing a child list specifying index values of zero or more nodes which are children of the node;

parsing an attribute list specifying zero or ~~more (attribute name, attribute value) pair~~ more attribute pair references for attributes of the node, each attribute pair reference comprising an attribute name and an attribute value; and

parsing a node value specification, which is empty if the node has no value; and

parsing a data buffer containing attribute names and attribute values referenced from the attribute lists and node values referenced from the node value specifications; and

using the parsed document as input for the processing.

Claim 34 (currently amended): A method for converting an input document encoded in an extensible human-friendly extensible markup language ("XML") to an output document encoded in a machine-oriented extensible markup language ("mXML"), comprising the steps of:

creating a document tree representation of the input document;

obtaining a node count representing a count of nodes in the document tree representation;

writing the node count to an mXML buffer;

traversing each node in the document tree representation and generating a corresponding node specification in the mXML buffer, further comprising the steps of:

generating a node name;

generating an attribute list specifying zero or ~~more (attribute name, attribute value) pair~~ more attribute pair references for attributes of the node, each attribute pair reference comprising an attribute name and an attribute value;

generating a child list specifying index values of zero or more nodes which are children of the node; and

generating a node value specification, which is empty if the node has no value;

generating a data buffer containing attribute names and attribute values referenced from the attribute lists and node values referenced from the node value specifications; and

appending the data buffer to the mXML buffer to form the output document.

Claim 35 (currently amended): The method according to Claim 34, wherein the step of generating ~~each (attribute name, attribute value) pair~~ each attribute pair reference further comprises the step of generating a starting name position, a name length, a starting value position,

and a value length.

Claim 36 (original): The method according to Claim 35, wherein the starting name position and starting value position are relative to a beginning of the data buffer.

Claim 37 (original): The method according to Claim 35, wherein the starting name position and starting value position are relative to a beginning of the output document.

Claim 38 (original): The method according to Claim 34, wherein the node value specification specifies a starting value position and a value length.

Claim 39 (original): The method according to Claim 37, wherein the starting value position is relative to a beginning of the data buffer.

Claim 40 (currently amended): The method according to Claim 37, wherein the ~~starting name position and~~ starting value position [[are]] is relative to a beginning of the output document.

Claim 41 (currently amended): The method according to Claim 34, wherein the step of generating ~~each (attribute name, attribute value) pair~~ each attribute pair reference further comprises the step of generating a starting name position, an ending name position, a starting value position, and an ending value position.

Claim 42 (original): The method according to Claim 34, wherein the node value specification specifies a starting value position and an ending value position.

Claim 43 (original): A document encoded in an extensible machine-oriented structured notation, wherein the document resides on one or more computer-readable media and comprises:

a node count representing a count of nodes in the document;

a node specification for each of the nodes, each of the node specifications comprising:

a node name;

a child list specifying index values of zero or more nodes which are children of the node; and

a node value specification, which is empty if the node has no value; and

a data buffer containing node values referenced from the node value specifications.

Claim 44 (original): A method for encoding a document in an extensible machine-oriented structured notation, comprising the steps of:

encoding a node count representing a count of nodes in the document;

encoding a node specification for each of the nodes, further comprising the steps of:

encoding a node name;

encoding a child list specifying index values of zero or more nodes which are children of the node; and

encoding a node value specification, which is empty if the node has no value;

encoding a data buffer containing node values referenced from the node value

specifications; and

storing the encoded node count, the encoded node specifications, and the encoded data buffer as the encoded document in memory or writing the encoded document to one or more storage media.